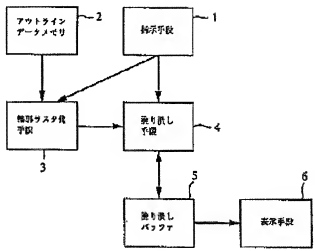
GRAPHIC PROCESSOR AND ITS METHOD

Publication number: JP7152920
Publication date: 1995-06-16
Inventor: KOBAYASHI YOSHIHISA; KOUNO MASAYOSHI
Applicant: VICTOR COMPANY OF JAPAN
Classification:
- international: ***G09G5/20; G06T11/40;*** **G09G5/20; G06T11/40;** (IPC1-7): G06T11/40; G09G5/20
- European:
Application number: JP19930325990 19931130
Priority number(s): JP19930325990 19931130

Abstract of **JP7152920**

PURPOSE:To solve an acute end and an overlap problem and to enable high- speed software processing. CONSTITUTION:An instruction means 1 instructs the kind, size, and modification of a graphic to be processed. Then outline data on the graphic instructed by the instruction means 1 are supplied from an outline data memory 2 to an outline raster scanning means 3. The outline data are so set that the direction of external loop data is opposite to the direction of internal loop data. Then the outline raster scanning means 3 makes a raster scan. For this raster scan, one outline point forming one outline is determined on one line and generated in a painting-out buffer 5. Further, a painting-out means 4 performs an additive subtractive painting-out process and the value of respective pixels of the painting-out buffer 5 are increased or decreased; when the value becomes larger than 1, the pixels are painted out on each line. Lastly, the graphic stored in the painting-out buffer 5 is displayed at a display means 6.



Family list
2 family member for: **JP7152920**
Derived from 1 application

Back to JP715

1 **GRAPHIC PROCESSOR AND ITS METHOD**
Inventor: KOBAYASHI YOSHIHISA; KOUNO MASAYOSHI
Applicant: VICTOR COMPANY OF JAPAN
EC:
IPC: ***G09G5/20; G06T11/40***; **G09G5/20** (+3)
Publication info: **JP2888270B2 B2** - 1999-05-10
JP7152920 A - 1995-06-16

Data supplied from the ***esp@cenet*** database - Worldwide

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報(A)　(11)特許出願公開番号

特開平7－152920

(43)公開日　平成7年(1995)6月16日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｔ　11/40				
Ｇ０９Ｇ　5/20		9471－5G		
		9365－5L	Ｇ０６Ｆ　15/ 72　４００	

審査請求　未請求　請求項の数４　ＦＤ　(全 13 頁)

(21)出願番号　特願平5－325990

(22)出願日　平成５年(1993)11月30日

(71)出願人　000004329
日本ビクター株式会社
神奈川県横浜市神奈川区守屋町３丁目12番地

(72)発明者　小林　芳尚
神奈川県横浜市神奈川区守屋町３丁目12番地　日本ビクター株式会社内

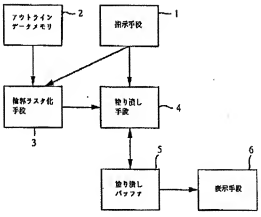
(72)発明者　河野　真儀
神奈川県横浜市神奈川区守屋町３丁目12番地　日本ビクター株式会社内

(54)【発明の名称】　図形処理装置及びその方法

(57)【要約】

【目的】　尖鋭端及び重複問題を解決し、また、高速ソフトウエア処理を可能にする。

【構成】　指示手段１は、処理を行う図形の種類、大きさ及び修飾を指示するものである。そして、この指示手段１よって指示された図形のアウトラインデータが、アウトラインデータメモリ２から輪郭ラスタ化手段３に供給される。アウトラインデータは、外ループデータの方向と内ループデータの方向とが逆方向となるように設定されている。そして、輪郭ラスタ化手段３によってラスタ化する。このラスタ化は、１つの輪郭線を形成する輪郭点を１つのラインに１つ決定して、塗り潰しバッファ５に作成する。さらに、塗り潰し手段４により、加減算塗り潰しを行って、塗り潰しバッファ５の各ピクセルの値をインクリメントまたはディクリメントし、値が１以上となったピクセルを１ラインごとに塗り潰す。最後に、塗り潰しバッファ５に記憶された図形を表示手段６に表示する。



1

【特許請求の範囲】

【請求項1】少なくとも図形の種類や大きさを指示する指示手段と、

ループ内が図形となる外輪郭を示す外ループデータの方向とループ外が図形となる内輪郭を示す内ループデータの方向とが逆方向となるように設定されたアウトラインデータを記憶するアウトラインデータ記憶手段と、

前記指示手段によって指示された図形の前記アウトラインデータが前記アウトラインデータ記憶手段から供給され、前記アウトラインデータの示す輪郭線を1つのラインごとに1つの輪郭点を決定してラスタ化する輪郭ラスタ化手段と、

この輪郭ラスタ化手段によって決定される前記輪郭線がライン処理方向と垂直方向に対して増加する方向か減少する方向かによって前記輪郭線で区切られる予め定められた一方向側の各ピクセルの値を1ラインごとにインクリメントまたはディクリメントする塗り潰し手段と、

この塗り潰し手段によって塗り潰される図形を記憶する塗り潰しバッファとを備えたことを特徴とする図形処理装置。

【請求項2】図形の輪郭を示すアウトラインデータをビットマップデータに変換する際にこの図形の内部または外部を塗り潰す図形処理方法であって、

ループ内が図形となる外輪郭を示す外ループデータの方向とループ外が図形となる内輪郭を示す内ループデータの方向とが逆方向となるように設定された前記アウトラインデータの示す輪郭線を1つのラインごとに1つの輪郭点を決定してラスタ化し、

前記輪郭線がライン処理方向と垂直方向に対して増加する方向か減少する方向かによって前記輪郭線で区切られる予め定められた一方向側の各ピクセルの値を1ラインごとにインクリメントまたはディクリメントするようにしたことを特徴とする図形処理方法。

【請求項3】前求項1記載の図形処理装置に使用される図形処理方法であって、

1ピクセルのデータを複数ビットの符号つき整数またはオフセットつきの符号なし整数で表現し、塗り潰し手段の1データワードに複数ピクセル分のデータを格納して、この1データワード単位またはその整数倍を単位としてインクリメントまたはディクリメント処理することを特徴とする図形処理方法。

【請求項4】請求項3記載の図形処理方法であって、

塗り潰し開始点に相当する複数ビット部分よりライン処理方向の逆方向に相当する複数ビット部分をオールセットにし、塗り潰し開始点に相当する複数ビット部分とその複数ビット部分よりライン処理方向の順方向に相当する複数ビット部分のそれぞれの最下位ビットをセットにしてそれ以外の各ビットをクリアにした加算値を加算することにより、インクリメント操作を行い、

塗り潰し開始点に相当する複数ビット部分よりライン処

2

理方向の逆方向に相当する複数ビット部分をオールセットにし、塗り潰し開始点に相当する複数ビット部分がデータワードの最下位にない場合はその最下位ビットをクリアにしてそれ以外の各ビットをセットにし、塗り潰し開始点に相当する複数ビット部分よりライン処理方向の順方向に相当する複数ビット部分がデータワードの最下位にない場合はその最下位ビットをクリアにしてそれ以外の各ビットをセットにし、最下位の複数ビット部分をオールセットにした加算値を加算することにより、デクリメント操作を行なうことを特徴とする図形処理方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、図形処理装置及びその方法に係り、特に文字などの図形の輪郭を示すアウトラインベクトルデータをドットデータに変換する際に、輪郭および図形の内部や外部を塗り潰す図形処理装置及びその方法に関するものである。

【0002】

【従来の技術】従来より、拡大・縮小した際にも高品質な文字として表示するために、文字の輪郭をアウトラインデータとして保持し、このアウトラインデータをドットデータに変換することによって描画される文字の輪郭の内部または外部を塗り潰す方法が採られている。

【0003】そして、コード化された文字を拡大・縮小してブラウン管等の出力装置に表示させる方法として、特公昭53-15624号の「文字等発生方式」や特公昭53-41017号の「高品位文字等の発生方式」等が知られている。ところが、これらの方法は、文字の輪郭の尖鋭端の部分が1ドットとなる場合に文字の塗り潰しが正しく行われないという問題があり、以降の特許出願等において、この尖鋭端問題を解決する方法がいくつか提案されている。

【0004】尖鋭端問題を解決して文字や図形のアウトラインベクトルデータを塗り潰す方法としては、特開平2-59872号の「画像処理装置」や特開平4-684号の「図形塗り潰し方法」、特開平3-233689号の「アウトラインデータ描画装置」等が提案されている。そして、特開平2-59872号の「画像処理装置」は、尖鋭端問題をソフトウエア処理に適した簡単な方法で解決する比較的優れた方法であり、輪郭線をプロットしながら、順次ラスタスキャン方向に反転塗り潰しを行なうことにより、尖鋭端問題を回避しつつ簡単な構成で塗り潰しを行なうようにしたものである。しかしながら、1スキャンラインについて同一の図形に多数回の反転処理が必要であるので、処理速度が低下するという問題がある。そこで、この提案では、図形の中心線近辺で塗り潰し処理領域を二分して、各領域での塗り潰し方向を変えることにより、反転処理が局部的に重複する度合いを軽減させているが、この場合でも処理速度が低下するという課題は残っていた。

【0005】また、塗り潰しの問題として、輪郭の重複

3　　4

問題がある。この重複問題をアウトライン文字の修飾の一つである文字の太さ変形を例にして以下に説明する。例えば、片仮名「ム」が図１０（Ａ）に示すようにアウトラインベクトルで定義されている場合、太さ変形を行なう方法の一例は、同図（Ｂ）に示すように、各頂点で接する二辺の成す角を二分する方向に、その頂点座標を移動することである。この時、同図（Ｂ）中、太い矢印で示した部分のように輪郭線の衝突あるいは重複が発生することがあり、この場合、特開平2-59872 号の方法では次のような問題が発生する。

【０００６】即ち、この輪郭線の重複が発生した部分を拡大したものを図１１（Ａ）～（Ｅ）に示し、輪郭線で囲まれた文字の内部を塗り潰す場合の反転塗り潰しの手順を説明すると、まず、同図（Ａ）に示すように、一番左側の輪郭線よりも右側を黒く反転させ、次に、同図（Ｂ）のように、その隣の輪郭線の右側を白く反転させて輪郭線で囲まれた部分だけを黒くする。ところが同様の処理を行っていくと、同図（Ｃ），（Ｄ）のように、輪郭線に囲まれた部分のうち、重複した部分を再度反転してしまうため、同図（Ｅ）のように、輪郭線に囲まれているにもかかわらず、白い中抜け部分が発生してしまうことがあった。

【０００７】特開平4-684 号の「図形塗り潰し方法」は、ピクセル（画素）を、輪郭フラグと方向ビットとの組み合わせビットで表現し、輪郭描画時と塗り潰し時とでこのピクセルを参照しながら書き換えを行なうことにより、拡大縮小に起因する重複問題を解決しようとするものである（なお、この方法では輪郭描画と塗り潰しは個別に行なっている）。特に、図１２に示されたような文字縮小に伴う、平行する輪郭の縮退による字画の消失に対しても対応することができるものである。

【０００８】ところが、この方法で輪郭描画を行う場合、輪郭描画開始後、ピクセルの方向情報の更新規則から漏れるケースが必ず発生する。例えば、この更新規則は前と次のピクセルのデータによって現在のピクセルに対する方向値が定まるため、輪郭の始点直後の処理は定まらないことになる（図１３（Ａ））。また、輪郭の始点直後からＹ座標値に変化のない点が続いた場合、不定点が連続して発生するようなことが起こりうる（同図（Ｂ））。したがって、特開平4-684 号に開示されている手順のほかに、輪郭描画一周後に、不定点を探して改めて決定し直す処理が必要であった。

【０００９】さらに、最も重要な方向情報の作成から塗り潰しに至る処理については、この提案には明確に記載されていないが、この場合、図１４（Ａ）に示すような内部に塗り潰しを行わない部分を有する図形に対して同図（Ｂ）に示すようなベクトルデータが存在するとき、更新規則による方向決定と不定点の再決定を行なって輪郭描画を行なうと図１６に示すようになる。なお、黒丸は輪郭ビットがセットされた点であり、数値の書き込まれていない点は、値が初期状態（＝０）の点である。

【００１０】そこで、水平方向にピクセル毎の値を加算して行くと、図１７に示すような結果が得られる。これに対して、①ゼロ②非ゼロ③偶数④奇数の条件判断を行ない、輪郭情報と論理和をとるなどの処理をすることにより、図１５（Ａ）に示す図形に対して、それぞれ図１５（Ｂ）～（Ｅ）に示すような結果が得られる。このとき、図１４（Ａ）に示すように、内部に塗り潰しを行わない部分を有する図形の塗り潰しを正確に行うためには、図１５（Ｅ）で示された結果が得られる④奇数の条件を採用することになる。

【００１１】ここで、太さ変形の例で説明したような輪郭線の重複が発生した際の問題点について説明する。図１８（Ａ）に示すような図形に対して同図（Ｂ）に示すようなベクトルデータがあったとすると（太さ変更に起因する状況を模式的に示した図である），先の例と同様に更新規則による方向決定と不定点の再決定を行なって輪郭描画を行なうと図１９に示すようになる。

【００１２】そして、水平方向にピクセルごとに値を加算して行くと、図２０に示すような結果が得られる。ここで上と同じように採用する条件を④奇数であることにすると、重複部分の値が－２となって奇数とならないので、重複部分で抜けが生じてしまう。他方、④非ゼロを条件として採用すると、本来塗り潰す必要のない内ループの内側まで塗り潰されることになり、本質的な問題が生じてしまう。さらに、指摘した不定点処理の問題も含め、この提案は、特殊ルールに基づく判断処理が多く、また、特殊な処理への分岐も多いため、この提案の実施例で示されたような構成の装置を用いてソフトウエア処理を行った場合、非常に処理時間のかかるものとなってしまう。

【００１３】そして、特開平3-233689号の「アウトラインデータ描画装置」は、上記した特開平2-59872 号と類似した特殊ルールに基づいた輪郭描画と塗り潰しを個別に行なう方法であり、上記提案で問題となった、輪郭線の重複部分の中抜け問題に対する対策について詳細に説明されているが、その使用されるルールが状態遷移を導入したより複雑なものになっている。このため、ハードウエア高速化を目的の一つにしているものの状態遷移処理等、現在使用されているＭＰＵ（microprocessing unit）によってソフトウエア処理を行うには、さらに時間のかかる方法となっていた。

【００１４】

【発明が解決しようとする課題】輪郭描画と塗り潰しを別々に行なう方法では、輪郭を単なるドットの集合としてではなく、輪郭の個々の部分の持つ性質を何らかの形で記憶することによって尖鋭端に対処する必要があった。一方、特開平2-59872 号のような方法では、尖鋭端問題は回避できるが、重複問題を避けることができなかった。

5

【0015】さらに、従来の方法では、尖鋭端問題と重複問題の両方を回避しようとすると、ソフトウエアの処理に時間が掛かり、あまり実用的ではなかった。そこで本発明は、上記した尖鋭端および輪郭重複に起因する問題を簡単な方法で解決し、MPUによるソフトウエア処理を行う場合に高速処理が可能な図形処理装置及びその方法を提供することを目的とする。

【0016】

【課題を解決するための手段】上記目的を達成するための手段として、少なくとも図形の種類や大きさを指示する指示手段と、ループ内が図形となる外輪郭を示す外ループデータの方向とループ外が図形となる内輪郭を示す内ループデータの方向とが逆方向となるように設定されたアウトラインデータを記憶するアウトラインデータ記憶手段と、前記指示手段によって指示された図形の前記アウトラインデータが前記アウトラインデータ記憶手段から供給され、前記アウトラインデータの示す輪郭線を1つのラインごとに1つの輪郭点を決定してラスタ化する輪郭ラスタ化手段と、この輪郭ラスタ化手段によって決定される前記輪郭線がライン処理方向と垂直方向に対して増加する方向か減少する方向かによって前記輪郭線で区切られる予め定められた一方向側の各ピクセルの値を1ラインごとにインクリメントまたはディクリメントする塗り潰し手段と、この塗り潰し手段によって塗り潰される図形を記憶する塗り潰しバッファとを備えたことを特徴とする図形処理装置、及び、図形の輪郭を示すアウトラインデータをビットマップデータに変換する際にこの図形の内部または外部を塗り潰す図形処理方法であって、ループ内が図形となる外輪郭を示す外ループデータの方向とループ外が図形となる内輪郭を示す内ループデータの方向とが逆方向となるように設定された前記アウトラインデータの示す輪郭線を1つのラインごとに1つの輪郭点を決定してラスタ化し、前記輪郭線がライン処理方向と垂直方向に対して増加する方向か減少する方向かによって前記輪郭線で区切られる予め定められた一方向側の各ピクセルの値を1ラインごとにインクリメントまたはディクリメントするようにしたことを特徴とする図形処理方法を提供しようとするものである。

【0017】

【作用】輪郭描画と塗り潰しを並行して行なうことにより、特別に輪郭の性質を記憶することなく、尖鋭端問題に対処する。また、アウトラインデータに記憶されている外ループデータと内ループデータとをループの向きで明確に区別し、輪郭点ごとに1ラインの加減算処理を行なって、各ピクセルの値の正負から塗りつぶされたか否かを判断することにより、重複部分の中抜けを防止することができる。さらに、MPUの処理しやすいワード長に複数ピクセルをまとめた状態で、特殊な判断を用いずに簡単なアルゴリズムを使用して処理することにより、処理速度を向上させることができる。

6

【0018】

【実施例】本発明の図形処理装置及びその方法の一実施例を図面と共に説明する。図1は、本発明の図形処理装置の一実施例を示す構成図である。同図において、指示手段1は、処理を行う図形の種類（どの図形を処理するか）、拡大縮小等の大きさ及び太さや形状等の修飾を指示するものである。そして、この指示手段1によって指示された図形のアウトラインデータが、アウトラインデータメモリ（アウトラインデータ記憶手段）2から輪郭ラスタ化手段3に供給される。アウトラインデータメモリ2に記憶されている各図形のアウトラインデータは、ループ内を図形として判断する外輪郭を示す外ループデータの方向とループ外を図形として判断する内輪郭を示す内ループデータの方向とが逆方向となるようにして頂点のデータと線の種類のデータがそれぞれ配列されたものである。そして、輪郭ラスタ化手段3によってこのアウトラインデータをラスタ化することによって輪郭を作成する。このラスタ化は、後述するように、各頂点を結ぶ輪郭線を形成する輪郭点を（塗り潰しバッファ5の）1つのラインごとに1つづつ決定するようにして行い、塗り潰し手段4を介して塗り潰しバッファ5に描画する。さらに、塗り潰し手段4により、後述する加減算塗り潰しを行って、塗り潰しバッファ5に描画されている図形の各ピクセル（画素）の値をインクリメントまたはディクリメントし、一定値以下または以上となったピクセルを塗り潰されたピクセルとすることにより、図形の塗り潰しを行う。最後に、塗り潰しバッファ5に記憶された図形を表示手段6に表示することにより、指示手段1にて指示された図形を表示することができる。

【0019】また、図2に本発明の図形処理装置の具体例を示す。図1に示した構成図と対比しながら説明すると、キーボード1aは、処理を行う図形の種類、大きさ及び修飾を指示する指示手段1に相当するものであり、MPU（microprocessing unit）7は、プログラムROM8に記憶されているプログラムに従って、アウトラインデータROM2から供給されるアウトラインデータに対して演算を行い、VRAM等の塗り潰しバッファRAM5aに出力して図形の描画と塗り潰しを行う回路である。なお、アウトラインデータROM2aは図1のアウトラインデータメモリ2に相当し、MPU7とプログラムROM8とが輪郭ラスタ化手段3と塗り潰し手段4とに相当する。そして、塗り潰しバッファRAM5aは塗り潰しバッファ5に相当している。塗り潰しバッファRAM5aに記憶された図形は、DA変換器9によりアナログビデオ信号に変換されて、CRT等の表示手段6に出力される。

【0020】次に、本発明の主要部分である図形の輪郭のラスタ化及び塗り潰しの方法について具体的に説明する。なお、図3以降の図面は、塗り潰しバッファ5内に生成される2次元イメージ平面をXY座標で表し、X座

標の増加方向を塗り潰し処理の進む方向とし、Y座標の増加方向が文字または図形の上方向、X座標の増加方向が文字または図形の右方向として表したものである。また、アウトラインデータメモリ2に記憶され、塗り潰しの対象となる文字の輪郭を示すアウトラインデータは、直線や近似曲線等で表現された一般的なアウトラインデータで良いが、そのデータの配列方向は、外ループと内ループとで逆向きに表現して、これらを明確に区別する必要がある。なお、一般にアウトラインデータは、展開処理系と対応づけられているので、このようなデータとしても問題は生じない。

【0021】（1）まず、輪郭の作成について説明する。アウトラインデータから輪郭を作成するためには、一般にDDA（Digital Differential Analyzer）アルゴリズムと呼ばれているアルゴリズムを使用する。このDDAアルゴリズムについては、各種国内の公知文献に記載されており、例えば、「実践コンピュータグラフィックス―基礎手続きと応用―」山口富士夫監修に詳しく紹介されている。DDAアルゴリズムは、直線をラスタ化するために、その直線を示す微分方程式の解を漸化式の形で表現し、XまたはYの値を1ピクセルづつ増加させながら、輪郭点を決定するものである。そして、DDAアルゴリズムのなかでも整数型Bresenham アルゴリズムは、少数値を使用せず、しかも除算を行わないため一般的なデジタル処理系で扱うのに都合が良い。また、誤差の少ない直線イメージのプロットが行なえるため、広く用いられている。

【0022】このBresenham 一般化アルゴリズムは、CRT等に描画された直線の輝度が、直線の向きによらず視覚的に一定となるように考慮し、一般化されたものである。そして、直線の向きをXY平面上の八分円で考え、第2と第3、第4と第5、第6と第7、第8と第1のそれぞれの八分円の組について、①アルゴリズムの変数としてXを選択するのか、②Yを選択するのか、それぞれ選択した変数を③インクリメントするのか、④デクリメントするのか、の4種類を対応づけるものである。

【0023】本実施例では、Bresenham アルゴリズムを使用するが、その目的は、描画された直線の輝度が、直線の向きによらず視覚的に一定となるようにすることではなく、塗り潰し処理の開始点を決定することである。そして、4種類全ての処理を考えると、1回に決まる塗り潰し開始点の数が、塗り潰し処理方向から見て一つに定まらず、不都合が生じるので、本実施例では、直線ベクトルの方向にかかわらず、Yのみを変数としてインクリメントのみ行なう（整数型Bresenham アルゴリズムの第2と第3八分円の組に対応する処理内容）ことにより、開始点を定めている。

【0024】ここで、DDAアルゴリズムと塗り潰しとの関連について説明する。図3（A）は整数型Bresenham 一般化アルゴリズムの第1八分円に対応する処理結果の例であり、同図（B）は第2八分円に対する処理結果の例である。そして、同図（A）では X 正方向に1つづつ増加させながら輪郭点を決定し、同図（B）では、Y正方向に1つづつ増加させながら輪郭点を決定したものとなっている。このようにして決定した輪郭点を塗り潰し処理の開始点としてX正方向に塗り潰し処理を行なう場合、同図（A）では、単一の輪郭線を形成する一つのY座標値（同一ライン）に対して、複数の輪郭点が存在するので、塗り潰し開始点が複数存在することになって、不都合が生じる。この結果、処理が重複するだけでなく、反転や加減算の結果に誤りを生じる虞れがある。

【0025】これに対して、同図（B）では、単一の輪郭線を形成する一つのY座標値に対しては、一つの輪郭点しか存在しないので、塗り潰し開始点も一つに決定され、不都合は生じない。そこで、本実施例では、アウトラインデータがどの分円に当てはまるかの判断は行なわず、第2と第3八分円の組に対応させて、変数がY座標のみとなるように、DDAアルゴリズムを変形して使用する。なお、このように対応させると輝度は一定となるとは限らないが、処理の結果得られる輪郭点は、塗り潰しの開始点として用いるためのものなので、輪郭の輝度等の変化は問題とならない。

【0026】（2）次に、塗り潰しについて尖鋭端問題を中心に説明する。本実施例の塗り潰し方法は、変形DDAアルゴリズムを使用して1つのラインごとに1つの輪郭点を決定してラスタ化することにより輪郭線を描画し、この輪郭線がライン処理方向と垂直方向に対して増加する方向であるときに輪郭線によって区切られた右側の各ピクセルの値を1ラインごとにインクリメント（増加）し、減少する方向であるときにデクリメント（減少）して、その値が1以上となったピクセルを塗り潰すピクセルとしたものである。

【0027】具体的には、まず、各ピクセルを数ビットの整数値で表現し、初期値を0とする。次に、変形DDAアルゴリズムにおいて、塗り潰し開始点が一つ決まるごとに、X座標値の増加方向に各ピクセル値の加減算を行なう。そして、変形DDAアルゴリズム処理を行なう際、もし、直線の始点のY座標値が終点のY座標値より小さければ、始点から終点に向かって塗り潰し開始点を決定して行く。そのとき、各ピクセルに対する操作はインクリメント（増加）である。もし、直線の始点のY座標値が終点のY座標値より大きければ、終点から始点に向って塗り潰し開始点を決定して行く。そのときの各ピクセルに対する操作は逆にデクリメント（減少）である。

【0028】次に、この方法で、尖鋭端を含む図形の塗り潰しを行なう場合について説明する。3点A（0，0）、B（4，8）、C（7，0）からなる三角形を考える。この三角形は外ループなので時計周りに定義される。まず、図4に示すように線分ABに対応する輪郭点

9

のプロットおよび塗り潰し処理を行なう。輪郭点は図中矢印の方向に、黒丸で示す各点に定まり、一つの輪郭点が決まるごとにその輪郭点とその輪郭点の右側すべての点(白丸の点) の値に加減算処理を行なう。この場合、線分がY座標増加方向なので、加算処理を行ない、右側各点の値は＋1となる。

【0029】次に、図5に示すように、線分BCに対応する輪郭点決定および塗り潰し処理を行なう。輪郭線は点Bから点Cの向きに定義されているが、逆転して矢印の方向（点Cから点Bへの方向）に描画する。輪郭点は黒三角の各点に決まり、一つの輪郭点が決まるごとにその輪郭点とその輪郭点の右側のすべての点(白三角の点) の値に加減算処理を行なう。この場合、線分をY座標増加方向に逆転したので、減算処理を行なう。その結果、各点の値は、図6に示す黒丸の点のみが＋1、その他の点は0となる。また、点Cと点AのY座標値は等しいので、線分CAに対しては処理を行う必要はない。

【0030】このようにして塗り潰しを行うと、尖鋭端を含む図形を塗り潰したときに尖鋭端が右側に尾を引くように塗り潰しが行われるなどの不都合が生じず、正確な塗り潰しを行うことができる。なお、ここでは、一線分に対するDDAアルゴリズムのループ条件を、Y変数＜終点のY座標値として例を示したが、条件をY変数≦終点のY座標値としても結果は同じになる。

【0031】（3）さらに、塗り潰しについて重複問題を中心に説明する。従来例にて使用した図10（A）に示すアウトラインベクトルで定義されている片仮名「ム」を同図（B）に示すように太くした場合に生じる図形の重複の処理について説明する。なお、同図（B）中、太い矢印で示した輪郭線の重複が発生した部分を拡大したものを図7（A）～（E）に示して説明する。ここでは、（2）で使用したのと同様、変形DDAアルゴリズムと加減算塗り潰しとを組み合わせた方法を使う。

【0032】まず、図7（A）に示す線分Aについて処理を行なうと、線分AはY座標値増加方向なので、この線分Aの左側は初期値0のままで、線分Aの右側は＋1となる。次に、同図（B）に示すように、線分Bについて処理を行なうと、線分BはY座標値減少方向なので、線分Aと線分Bの間は＋1のままで、線分Bの右側は差し引き0となる。

【0033】さらに、同図（C）に示すように、線分Cについて処理を行なうと、線分CはY座標値増加方向なので、線分Cの右側のビクセルをすべて＋1にする。その結果、線分Cの右側のうち線分Aと線分Bとで挟まれた部分（線分Cと線分Bとの間の部分）では＋2となり、それ以外の部分（線分B及び線分Cよりも右側の部分）では＋1となる。また、線分Cの左側のうち線分Aと線分Bとで挟まれた部分（線分C及び線分Bよりも左側の部分）は＋1のままであり、それ以外の部分（線分Bと線分Cとの間の部分）は0のままである。

10

【0034】そして、同図（D）に示すように、線分Dについて処理を行なうと、線分Cと同様に、Y座標値増加方向であることから線分Dの右側のビクセルをすべて＋1にし、図のように線分Dの右側のうち線分Aと線分Bとで挟まれた部分を＋2にして、それ以外の部分を＋1にする。最後に、同図（E）に示すように、線分Eについて処理を行なうと、線分EはY座標値減少方向なので、線分Eの右側のビクセルをすべて－1にして、その値は差し引き0となる。そして、このように処理して得られた値が正（1以上）であるビクセルを塗り潰すビクセルとすることにより、輪郭が重複した部分でも正しい塗り潰し結果を得ることができる。

【0035】したがって、変形DDAアルゴリズムと加減算塗り潰しとを組み合わせた方法を使うことにより、尖鋭端問題と重複問題の両方を解決することができる。また、上記実施例では、ループ内が図形となる外輪郭を示す外ループデータについて説明したが、ループ外が図形となる内輪郭を示す内ループデータの場合でも、そのデータの配列方向を外ループデータと逆方向となるようにアウトラインデータを保持しておくことにより、全く同様の処理で正しい塗り潰しを行うことができる。

【0036】（4）最後に、塗り潰しを高速化処理する方法について説明する。ここでは、特にMPUを使用したソフトウエア処理において高速化する方法について説明する。上記で説明した本発明の塗り潰し方法では、各ビクセルに加減算処理を行うために、複数ビットで1ビクセルを表現する必要がある。実際に何ビット必要になるかは文字図形の複雑さによるが、通常日本語アウトラインフォントに使用する場合は、4ビット程度で十分である。一方、一般的にMPUが扱うデータの処理単位は2のべき乗であり、比較的大量のデータ処理を行なう場合には、16ビットまたは32ビットのデータバス幅を持つMPUが通常使用される。

【0037】ここで、複数ビクセルを組み合わせて効率良く塗り潰し処理を行なう例として、16ビットアクセス可能なバッファメモリを使って、4ビット／ビクセルで表現されたデータの処理を考える。この場合、ビクセル配置とメモリ配置は図18（A），（B）に示すように対応けることができる。なお、同図（A）はバッファメモリ上に文字のイメージを表現したときのビクセル配置を示す図であり、同図（B）はMPUから見たバッファメモリのアドレス配置図である。そして、1ワードが16ビットのメモリに4ビットのビクセルを配置するので、1ワード内に4ビクセルが配置されることになる。

【0038】ここで、DDAアルゴリズム等で決定した塗り潰し開始点（輪郭点）は、メモリ配置上では、ビット15－12、ビット11－8、ビット7－4、およびビット3－0の4つの部分のいずれかに対応づけられたデータワードとなる。一方、塗り潰し開始点に対応づけ

られたデータワードよりビクセル配置上右側に対応するデータワードは、すべてのビット部分をインクリメントまたはデクリメント処理することになる。

【００３９】したがって、図９に示すように、開始点の塗り潰しは、上記４つの場合に分けられるが、ビクセル配置上その右側に対応するデータワードは、全てビット１５－１２に塗り潰し開始点が対応づけられた場合と同等の処理を行なえばよいことになる。そして、上記の４つの場合の対応づけの方法は、Ｘ軸方向１ライン分のビクセル数が、１データワードが表現するビクセル数（４）の整数倍である（図９に示すような場合）と仮定すれば、決定した塗り潰し開始点のＸ座標値を１データワードが表現するビクセル数で割った際の余りに応じて対応づけすることができる（図中、余りが０，１，２，３のとき、それぞれａ，ｂ，ｃ，ｄ…ａｎ，ｂｎ，ｃｎ，ｄｎの各行に対応する）。

【００４０】このように、４つの場合の対応づけを行って、塗り潰し開始点及びその右側のビクセルをインクリメントまたはデクリメントしていき、最終的に１以上（または０以下）となったビクセルを塗り潰されたとして処理すれば良い。

【００３６】次に、ビクセルの持つ値の実際の表現方法と実際の加減算方法について説明する。上記したように、ビクセルの持つ値を通常の整数表現で処理しても良いが、表現の方法を工夫することにより、ソフトウエアで処理する際に、上記の組み合わせ処理を効率的に行なうことができる。なお、この方法は、ハードウエアで構成した場合でも高速処理が可能である。通常は符号つきの整数を２進化表現する場合、２の補数表現で行なうことが多いが、オフセットを付けた上で符号なし２進化整数の形で表現することもできるので、ここでは、後者の符号なし２進化整数の形で表現する場合を例にとって説明する（なお、加減算する方の数値は、２の補数表現で行なう）。

【００４１】オフセット付き符号なし２進化整数の表現をｎビットで表現するときには、０の値を最上位ビット以外がオールセットとし、2^{n}／２の値をオールクリア、－（(2^{n}／２）－１）の値をオールセットとしたビット符号で表現する。そして、４ビット符号の例を示すと、次のようになる。

【００４２】

+8	1111
+7	1110
：	：
+1	1000
0	0111
−1	0110
：	：
−7	0000

【００４３】この値に対して加減算（インクリメントとデクリメント）を行うには、

インクリメント：　+0001

デクリメント：　+1111（ただし、キャリーを無視）

の演算を行なうことで成立する（なお、以下に示す演算は、被加算値を２の補数表現にした場合でも加算値は同じである）。

【００４４】この演算処理を複数ビクセルを組み合わせて処理する場合について図８（Ａ），（Ｂ）及び図９の場合を例にして説明する。４ビクセルの組み合わせに対してインクリメントする場合は、上述したように塗り潰し開始点に相当するビット部分の位置によって、加算値は下記４つの場合に分けられる。

【００４５】

塗り潰し開始点　加算値（2進表現16ビット）

ビット15-12：　0001　0001　0001　0001

ビット11-8：　0000　0001　0001　0001

ビット 7-4：　0000　0000　0001　0001

ビット 3-0：　0000　0000　0000　0001

【００４６】一方、デクリメントする場合は、キャリーが上位のビットに伝搬するので、この処理に注意する必要がある。例えば、初期化されている２ビクセルを両方ともデクリメントする場合には、

初期値：±0　0111　0111

加算値：－1　1110　キャリ　1111

加算結果：　1000　←　1000

となるように、最下位に配置されたビクセル部分以外は、加算値がオールセット（２の補数表現の－１）ではなく、最下位ビットはクリア（２の補数表現の－２）とする必要がある。

【００４７】さらに、塗り潰し開始点に相当するビット部分の位置が最上位にない場合は、開始点より左側に相当する（デクリメントしない）ビット部分の加算値は、キャリーを相殺するように、オールセット（２の補数表現の－１）である必要がある。例えば、初期化されている２ビクセルのうち、下位のビクセルのみをデクリメントするときは、

初期値：±0，±0　0111　0111

加算値：±0，－１　1111　キャリ　1111

加算結果：　0111　←　1000

のようにする。

【００４８】したがって、４ビクセルの組み合わせに対してデクリメントする場合は、塗り潰し開始点に相当するビット部分の位置によって、加算値は下記４つの場合に分けられ、それぞれに対応させた加算値を用いることにより、正しいデクリメントが可能となる。

【００４９】

塗り潰し開始　　加算値（2進表現16ビット）

ビット15-12：1110　1110　1110　1111

ビット11-8　：1111　1110　1110　1111

ビット 7-4　：1111　1111　1110　1111

ビット 3-0 ：1111　1111　1111　1111

【0050】以上４ビセルの場合のインクリメントとデクリメントについて説明したが、本発明は、他のビクセル数及び３２ビットの場合などでも同様にして加算値を求めて加減算を行うことができ、上記したような加算値を加算するだけで、インクリメントやデクリメントを行うことができるので、高速演算処理が可能であり、さらに、最終的な値を得てから塗り潰すビクセルを決定すれば良いので、何度も反転処理を行う必要がなく、塗り潰し処理も高速で行うことができる。

【0051】

【発明の効果】本発明の図形処理装置及びその方法は、輪郭描画と塗り潰しを並行して行なっているので、特別に輪郭の性質を記憶することなく、尖鋭端の塗り潰しを正確に行うことができる。

【0052】また、アウトラインデータに記憶されている外ループデータの方向と内ループデータの方向とを逆方向にして明確に区別した上で、輪郭点ごとに１ラインの加減算処理を行なって、各ビクセルの値の正負から塗りつぶされたか否かを判断できるようにしているので、重複部分の中抜けを防止することができる。また、外ループデータと内ループデータとを区別することなく、同じ処理で正確な塗り潰しを行うことができる。

【0053】さらに、ＭＰＵの処理しやすいワード長に複数ビクセルをまとめた状態で、特殊な判断を用いずに簡単なアルゴリズムを使用して処理しているので、高速でのソフトウエア処理が可能となると共に、メモリを高効率で利用することができる。そして、ハードウエアで処理を行う場合も簡単な構成で実現することができる。

【0054】また、複数ビクセルを同時に扱うことにより、近年のＭＰＵに一部実装されているハード的なデータバス幅の複数倍のデータビット数を単一命令にて処理できるような機能を有効に利用することができるという効果がある。

【図面の簡単な説明】

【図１】本発明の図形処理装置の一実施例を示す構成図である。

【図２】本発明の図形処理装置の具体例を示す構成図である。

【図３】（Ａ）は整数型Bresenham 一般化アルゴリズムの第１八分円に対応する処理結果の例を示す図、（Ｂ）は同じく第２八分円に対応する処理結果の例を示す図である。

【図４】本発明で尖鋭端を含む図形の塗り潰しを説明するための図である。

【図５】本発明で尖鋭端を含む図形の塗り潰しを説明するための図である。

【図６】本発明で尖鋭端を含む図形の塗り潰しを説明するための図である。

【図７】（Ａ）～（Ｅ）は本発明で塗り潰しの重複問題を説明するための図である。

【図８】（Ａ）は本発明の一実施例でバッファメモリ上に文字のイメージを表現したときのビクセル配置を示す図、（Ｂ）はＭＰＵから見たバッファメモリのアドレス配置を示す図である。

【図９】本発明の一実施例で開始点の塗り潰しの場合分けについて説明するためのビクセル配置を示す図である。

【図１０】（Ａ）は文字の例を示す図、（Ｂ）はその文字の太さを変形した例を示す図である。

【図１１】（Ａ）～（Ｅ）は従来の塗り潰しの重複問題を説明するための図である。

【図１２】従来例を説明するための図である。

【図１３】（Ａ），（Ｂ）はそれぞれ従来例の問題点を説明するための図である。

【図１４】（Ａ）は図形の例を示す図、（Ｂ）はそのベクトル例を示す図である。

【図１５】（Ａ）は図形の例を示す図、（Ｂ）～（Ｅ）はその図形の塗り潰し結果の例を示す図である。

【図１６】図１４の図形の輪郭描画を行なった例を示す図である。

【図１７】図１４の図形の輪郭描画を行なった例を示す図である。

【図１８】（Ａ）は図形の例を示す図、（Ｂ）はそのベクトル例を示す図である。

【図１９】図１８の図形の輪郭描画を行なった例を示す図である。

【図２０】図１８の図形の輪郭描画を行なった例を示す図である。

【符号の説明】

1　指示手段

1ａ　キーボード

2　アウトラインデータメモリ（アウトラインデータ記憶手段）

2ａ　アウトラインデータＲＯＭ

3　輪郭ラスタ化手段

4　塗り潰し手段

5　塗り潰しバッファ

5ａ　塗り潰しバッファＲＡＭ

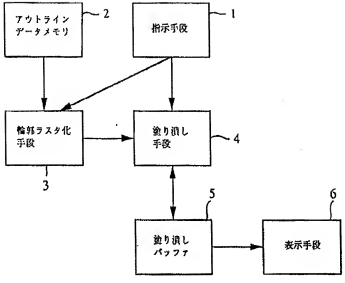
6　表示手段

7　ＭＰＵ（microprocessing unit）

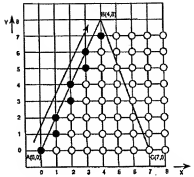
8　プログラムＲＯＭ

9　ＤＡ変換器

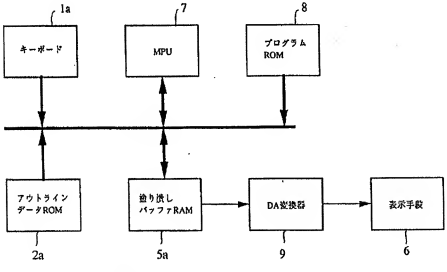
【図1】



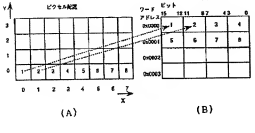
【図4】



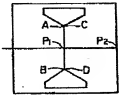
【図2】



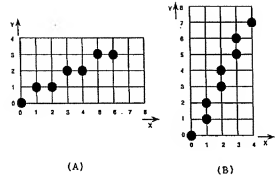
【図8】



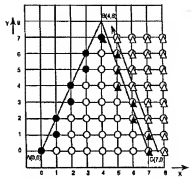
【図12】



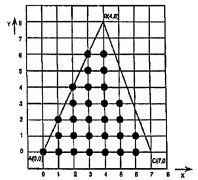
【図3】



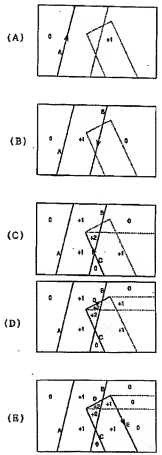
【図5】



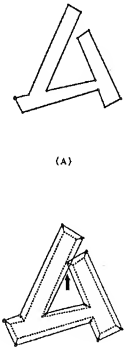
【図6】



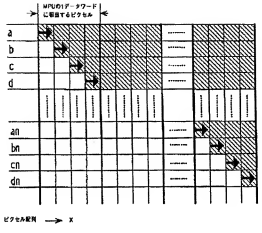
【図7】



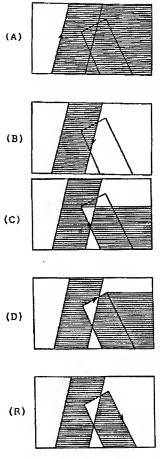
【図10】



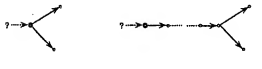
【図９】



【図１１】

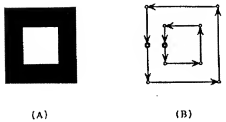


【図１３】



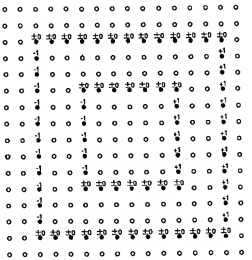
(A)　　(B)

【図１４】

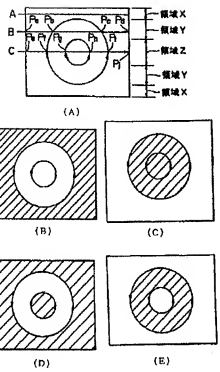


(A)　　(B)

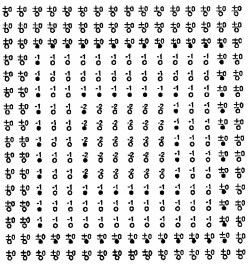
【図１６】



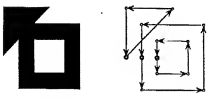
【図15】



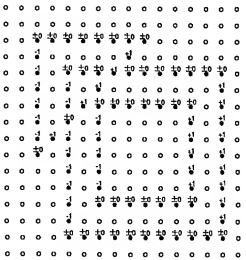
【図17】



【図18】



【図19】



【図２０】